20 VIDEOLEZIONI DI BASIC
PER IMPARARE COL VIC 20

GRUPPO
EDITORIALE
JACKSON

8

COMMODORE
VIC 20

VIDEO BASIC VIC 20
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea - Via Indipendenza 88 - Como
Redazione software:
Francesco Franceschini, Enrico Braglia,
Fabio Calanca
Segretaria di Redazione:
Marta Menegardo
Progetto grafico:
Studio Nuovaidea - Via Longhi 16 - Milano
Impaginazione:
Silvana Corbelli
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

Autorizzazione alla pubblicazione Tribunale di
Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70
(autorizzazione della Direzione Provinciale delle
PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo
Editoriale Jackson S.r.l. - Via Rosellini, 12
20124 Milano, mediante emissione di assegno
bancario o cartolina vaglia oppure
utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere
richiesti direttamente all'editore
inviando L. 10.000 cdu. mediante assegno
bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

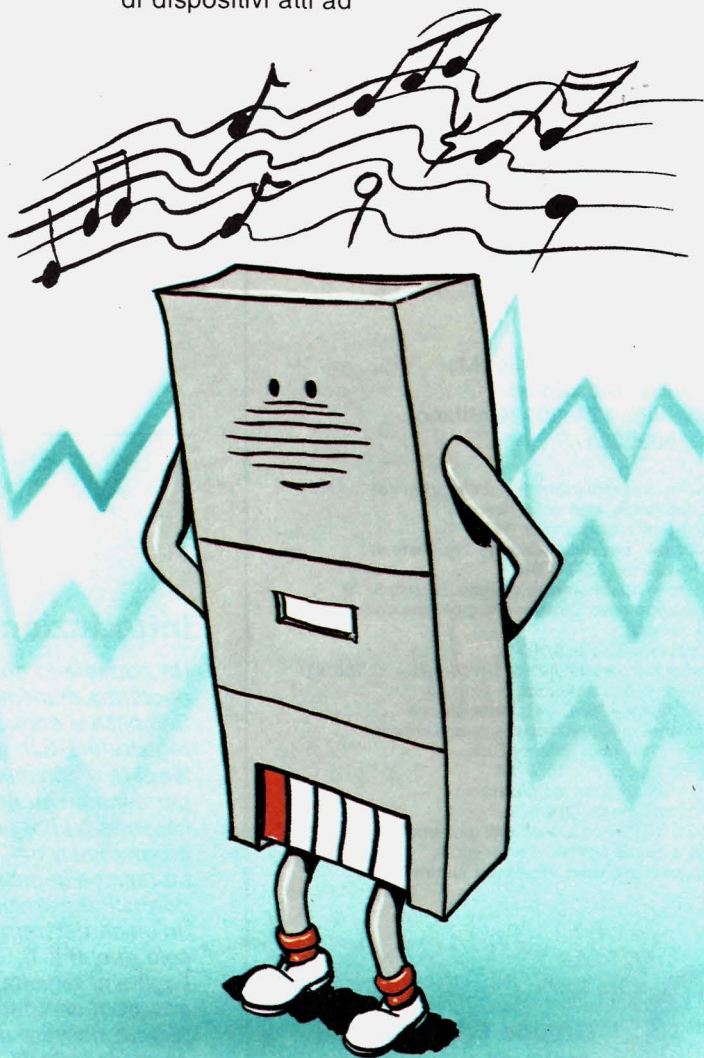
Gruppo Editoriale Jackson

SOMMARIO

Introduzione

Un normale (o quasi) registratore è la periferica di memoria per eccellenza. Semplice e economico, infatti, il registratore è in grado di scrivere, leggere e conservare in modo permanente su nastro (una comune cassetta audio) tutte le informazioni: programmi o dati che siano.
La tecnica impiegata è quella delle normali registrazioni audio, con un'unica differenza: le "note" sono solo due; 0 e 1.
Legate al registratore, poi, vi sono istruzioni che permettono il colloquio (dare o ricevere dati tra registrare e computer): SAVE, VERIFY e LOAD.

Il registratore

Un sistema per l'elaborazione dei dati richiede, come condizione indispensabile per il trattamento delle informazioni, la presenza di dispositivi atti ad immettere dati nel sistema e a registrarli in uscita. Tali funzioni - come ben sai - vengono espletate dalle cosiddette unità di ingresso/uscita,

collegate direttamente al sistema ed in grado di fornire o restituire, quando necessario, quanto occorre all'unità centrale per eseguire le varie operazioni.
Tra tutte le periferiche di I/O il registratore magnetico è il dispositivo di ingresso/uscita di più comune utilizzo, diffuso sia tra i piccoli che i grandi computer.
Abbiamo già detto che la memoria centrale (a lettura e scrittura, cioè RAM) risulta allo stato attuale ancora di costo piuttosto elevato (nonostante i notevoli progressi raggiunti negli ultimi tempi dalla tecnologia elettronica) fattore che generalmente costringe gli utenti a utilizzare RAM di capacità limitata. Inoltre, essa è quasi sempre volatile, cioè perde irrimediabilmente il proprio contenuto quando il computer viene spento. È perciò necessario che programmi e dati possano venir conservati su memorie di tipo non volatile, dette anche memorie di massa proprio perché in grado di contenere grandi quantità di dati e di informazioni.

I supporti magnetici

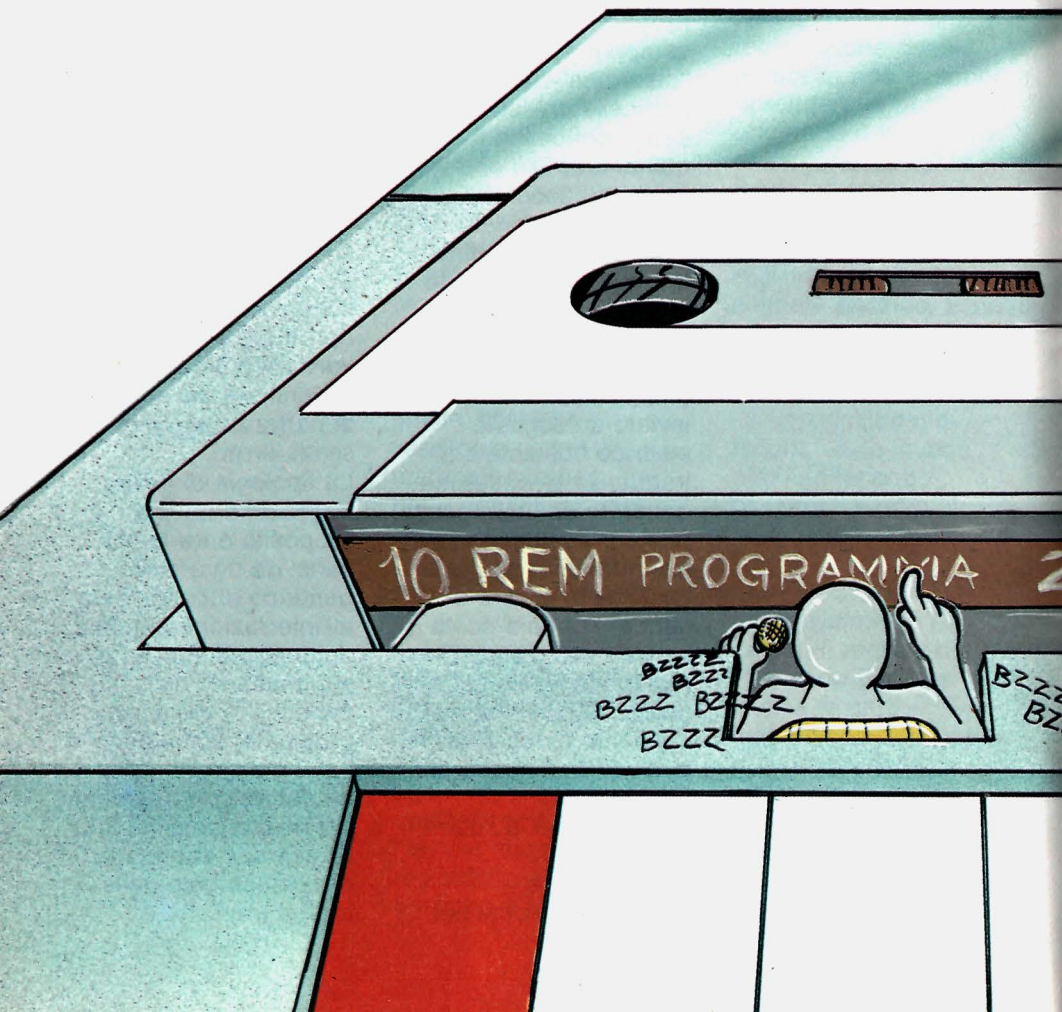
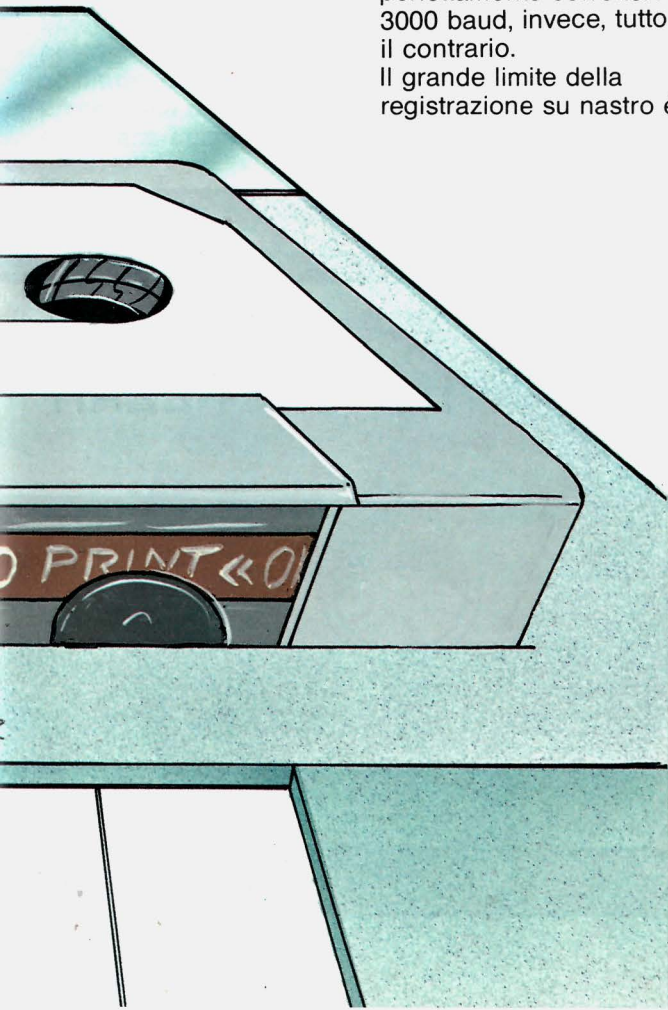
Il registratore - o, meglio, il nastro magnetico - è appunto una memoria di massa: esso rappresenta un supporto di registrazione fondamentale per gli elaboratori ed è usato, sia come unità di ingresso che di uscita, per memorizzare stabilmente programmi e risultati di calcolo o, più semplicemente, dati ed informazioni.
Le unità a nastro magnetico offrono infatti la possibilità di immettere dati nella memoria centrale ad una velocità sufficientemente elevata e di registrarli in uscita in modo efficiente, sicuro e - soprattutto - economico.
La tecnologia ed i principi di funzionamento sono identici per qualsiasi tipo di unità a nastro, a partire dai microregistratori per arrivare ai super-terminali a nastro dei grossi centri di calcolo.
Alla base di tutto si trova infatti la stessa filosofia costruttiva, per quanto applicata - com'è d'altra parte logico - su scale, proporzioni e tecnologie

alquanto diverse.
Nella nostra trattazione faremo esplicito e costante riferimento ai normali registratori a cassetta; alla luce di ciò che è stato appena detto il discorso è comunque applicabile e generalizzabile a qualunque altro genere di dispositivo a nastro magnetico.

Per prima cosa occupiamoci del principio su cui si basa la registrazione magnetica. Senza entrare in una dotta dissertazione sul magnetismo e l'elettromagnetismo, diciamo che alcuni materiali, detti ferromagnetici, hanno la proprietà di calamitarsi in modo permanente, se esposti ad un intenso campo magnetico.
Questo stato cessa (viene annullato o modificato) in presenza di un altro campo magnetico sufficientemente intenso.
E veniamo al nostro registratore. Il nastro magnetico usato è costituito da un supporto non magnetico - un sottile, flessibile e resistente nastro di materiale plastico - su cui è depositato uno strato estremamente fine di speciali sostanze magnetiche (ossidi di ferro o altro) assimilabili a numerose, microscopiche calamite, indipendenti una dall'altra.
Il nastro viene fatto scorrere a velocità costante sotto una "testina magnetica di registrazione", la cui funzione è quella di trasformare i segnali elettrici in ingresso in una serie di corrispondenti segnali magnetici.
Le variazioni di campo magnetico prodotte dalla testina si traducono in invisibili movimenti di orientamento delle singole "calamite" che compongono lo strato sensibile. Cessato l'effetto della testina di registrazione queste particelle non riescono a riprendere la posizione originaria e sul nastro rimangono quindi in modo permanente le informazioni magnetiche trasmesse in origine al registratore attraverso un segnale elettrico. È stata effettuata la registrazione, o scrittura, del nastro.
La riproduzione, o lettura, viene invece ottenuta facendo passare il nastro stesso sotto una testina di lettura, di uso e funzionamento sostanzialmente simile a quella di registrazione. Passando dinnanzi alla testina le parti magnetiche del nastro, provocano delle variazioni di campo magnetico uguali e contrarie a quelle che le avevano generate, determinando pertanto

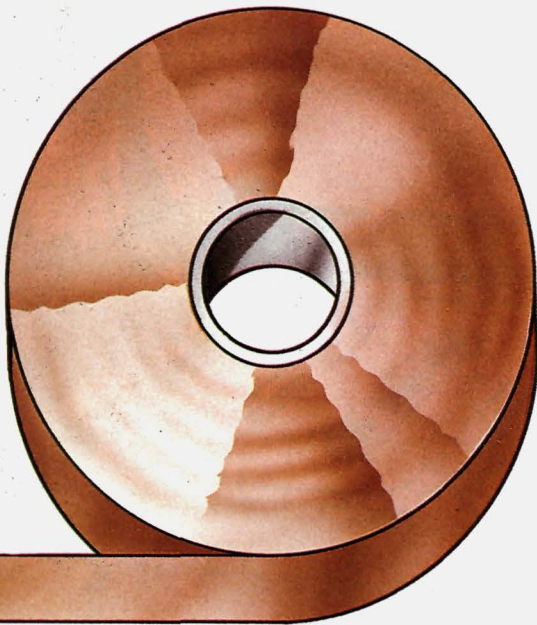
sulla testina che le percepisce l'insorgere di una serie di segnali elettrici molto deboli. Amplificandoli migliaia di volte si ottiene un segnale simile a quello registrato.
Una particolarità importante della registrazione magnetica è la possibilità della cancellazione rapida dell'informazione immagazzinata, cosa questa che consente di riutilizzare il nastro anche centinaia di volte senza provocare apprezzabili deterioramenti di qualità.
Il metodo più usato è quello di far passare il nastro magnetico sotto una "testina di cancellazione", che produce un intenso campo magnetico e "disordina" tutte le particelle presenti sullo strato, cancellando quindi tutto ciò che vi era in precedenza memorizzato.
Il funzionamento di tale testina è grosso modo simile a quello della testina di registrazione, con la sola differenza che ad essa viene inviato un segnale elettrico costante e di frequenza elevatissima, generato da un apposito circuito del registratore.
Nei normali registratori a cassette la testina di cancellazione è posta a fianco della testina di registrazione, in posizione tale che - al momento di registrare qualcosa - il nastro sia costretto, prima di essere inciso, a subire il processo di cancellazione, preparandosi così nel migliore dei modi alla successiva registrazione.
Naturalmente, in tutto il processo risultano di fondamentale importanza sia la velocità di trascinamento del nastro (che deve essere mantenuta il più possibile precisa, lineare e costante) sia la zona di contatto tra testina e nastro magnetico (che deve conservare una certa posizione di allineamento). Un registratore che non rispetti anche una sola di queste condizioni non potrà infatti mai svolgere correttamente il proprio lavoro, cioè memorizzare le informazioni e successivamente riprodurle senza introdurre alcuna distorsione del segnale di partenza e quindi senza errori.
La tipologia di errori segnalati a questo proposito è tra le più varie: da un errore generico di caricamento all'interruzione (a metà) del programma, al superamento della capacità di memoria ... Il problema, come detto, è il più delle volte dovuto a un cattivo trascinamento del nastro (velocità non costante) e soprattutto al disallineamento della testina (non

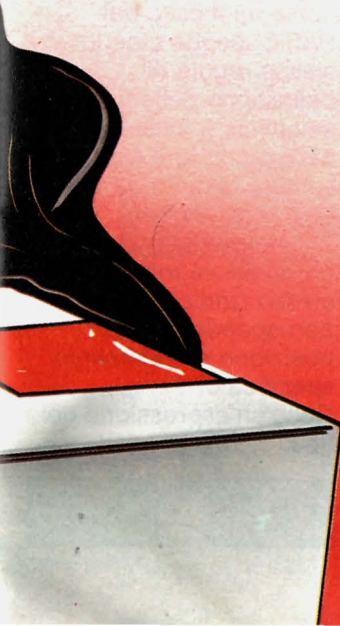
perpendicolare al nastro).
È sufficiente infatti anche una minima variazione perché la quantità e l'intensità del segnale in lettura diminuisca drasticamente.

Si tratta di un difetto abbastanza subdolo, in quanto non è avvertibile con dati salvati mediante lo stesso registratore.
Una testina disallineata registra infatti le informazioni in un modo diciamo "personale".
Così come le registra, però, è in grado di rileggerle, dando l'impressione di un buon funzionamento.
I problemi di registrazione, inoltre, aumentano con l'aumentare della velocità di trasferimento dei dati; più lenta, infatti, è questa, minore è la densità di informazione,

cioè ci sono meno dati per centimetro di nastro. Più alta è la velocità, maggiore è la densità e quindi più critica diventa la lettura/scrittura.
Ad esempio, con una velocità di 300 baud - la velocità di trasferimento dei dati viene espressa in BPS, bit per secondo, o BAUD - si ha una bassa densità di registrazione, un tempo molto lungo di registrazione o lettura, ma una buona probabilità di successo anche in condizioni di allineamento non perfettamente corrette. A 3000 baud, invece, tutto il contrario.
Il grande limite della registrazione su nastro è comunque insito nel supporto utilizzato: un nastro, infatti, per essere letto o registrato deve essere svolto dall'inizio alla fine.
L'utilizzo pertanto può avvenire solo in modo sequenziale, ossia scrivendo i dati e le informazioni uno dopo l'altro.
In fase di lettura essi verranno prelevati con la stessa sequenza; sarà possibile cercare i dati che interessano in fase di svolgimento del nastro, ma non sarà possibile prelevare le informazioni che già sono passate. Per recuperare un dato precedente bisognerà riavvolgere il nastro e ricominciare la ricerca dall'inizio.
Nonostante tutto ciò, il registratore è di uso comunissimo: la comodità e la semplicità di funzionamento, abbinate alla notevole economicità ed affidabilità, lo pongono infatti tra le unità di memorizzazione maggiormente tenute in considerazione.
Nelle prossime lezioni vedremo comunque che esistono dei dispositivi i quali permettono di risolvere tutte le limitazioni appena viste.

Header: i dati su nastro

Adesso che hai imparato le cose principali sul funzionamento del registratore possiamo tranquillamente passare alla seconda fase della nostra lezione, cioè alla parte relativa alla connessione ed al "colloquio" tra il tuo computer e, appunto, il registratore.
Cerchiamo innanzitutto di spiegare le modalità con cui gli elaboratori memorizzano e leggono i dati, le informazioni ed i programmi attraverso i nastri magnetici.
Come al solito, quando si realizza un collegamento tra l'unità centrale ed una periferica occorrono due cose indispensabili: un'interfaccia (che consenta alla CPU di comunicare con l'esterno) ed una connessione (che unisca fisicamente il computer alla periferica). Il registratore non sfugge certamente a questa regola. Comunque, al di là del collegamento fisico, è pure necessario che tra le due unità esista anche una perfetta intesa di comunicazione, cioè una regola per il colloquio e la ricezione delle informazioni in arrivo ed in partenza. La cosa sembra apparentemente complicata.
Vediamo di chiarire il tutto con un esempio, che ci mostra cosa

PUNTI MAGNETIZZATI

succede quando all'unità centrale viene impartito l'ordine di memorizzare un programma su nastro. La prima cosa che la CPU provvede a verificare è che il dispositivo ricevente, cioè il registratore, sia in grado di ascoltarla, controllando pertanto che tutto sia predisposto per l'arrivo delle informazioni. Se tutto è a posto, può quindi cominciare l'invio del programma da registrare. Tale operazione viene compiuta utilizzando l'apposita interfaccia, che trasmette i dati attraverso una connessione seriale, inviando perciò al registratore un bit dopo l'altro.
Perché seriale? La risposta è molto semplice. Il registratore memorizza le informazioni in arrivo in ordine sequenziale, cioè una di seguito all'altra. È pertanto naturale che venga scelto come standard di connessione e comunicazione quello che più si avvicina a questo modo di operare, cioè a questa caratteristica di sequenzialità.
Terminata la trasmissione, sul nastro si trova la versione "magnetica" del programma - ogni singolo bit è rappresentato dalla presenza/assenza di informazione magnetica - già pronta per essere riversata nuovamente, quando necessario, nella memoria del computer.
In tutto questo discorso è però stata (volutamente) tralasciata un'azione molto importante che la CPU esegue prima di memorizzare il programma: la preventiva operazione di scrittura sul nastro, all'inizio della registrazione, del cosiddetto header.
Un header è una sorta di presentazione introduttiva (header significa appunto intestazione) di ciò che immediatamente dopo verrà memorizzato; la registrazione di un programma è quindi affidata a due blocchi, separati tra loro da un breve spazio. Il primo è

un blocco preliminare, contenente il nome e qualche altra informazione sul programma; il secondo comprende invece il programma vero e proprio.
La presenza e la funzione dell'header, che comincia di solito con una nota continua, sono fondamentali: comunica all'unità centrale il nome del programma, la sua lunghezza ed altre informazioni, che consentono la sincronizzazione delle operazioni di salvataggio o di caricamento, nonché i controlli sul loro buon esito.
Il nome del programma non è generalmente obbligatorio, ma è consigliabile farne sempre uso per non rischiare di confondere informazioni diverse.
La registrazione dell'header viene comunque eseguita automaticamente dal computer (se ne incarica, infatti, il sistema operativo) durante il caricamento: nessun impegno supplementare è pertanto richiesto al programmatore.
Anzi, nei confronti dell'utente l'header è - come si dice - "trasparente", cioè invisibile: la sua presenza e la sua scrittura costituiscono pertanto argomenti che interessano e dipendono solo ed unicamente dal calcolatore.
L'unica cosa di cui occorre assicurarsi è, piuttosto, che le tacche di protezione di registrazione al margine della cassetta siano bloccate (abilitando cioè la memorizzazione) e che il nastro non sia completamente avvolto sulla cassetta stessa.

Il nastro è infatti di solito preceduto da un breve tratto non magnetizzato (immediatamente riconoscibile perché trasparente o colorato): cominciando la registrazione dall'inizio l'header verrebbe ad essere inviato proprio in corrispondenza di tale zona non registrabile, con l'ovvia conseguenza che il programma in seguito non potrebbe più essere recuperato, venendo a mancare alla CPU la parte di riconoscimento e sincronizzazione. Attenzione, quindi!
In commercio esistono comunque speciali cassette, il cui uso è espressamente rivolto ai computer, alle quali è stato eliminato questo tratto "bianco". Se proprio non si vogliono correre rischi ...
Per quel che riguarda la qualità della registrazione bisogna dire che - oltre naturalmente al registratore utilizzato - essa dipende anche dalla "grana" dello strato magnetico del nastro, cioè dalla dimensione media di ciascuna delle particelle magnetizzate che costituiscono il supporto. Quanto più la grana è fine, tanto più il nastro è infatti in grado di percepire le più rapide variazioni di campo magnetico.
A differenza però delle registrazioni audio (un orecchio percepisce frequenze tra 0 e 14.000 Hz) il computer è un po' "sordo": arriva al massimo a 3000 Hz.
Per ottenere delle buone incisioni non è quindi necessario spendere una fortuna in attrezzature e materiali: le uniche precauzioni da rispettare sono soltanto quelle di custodire i nastri e le cassette in luoghi freschi ed asciutti (e soprattutto lontano da sorgenti di forti campi magnetici, come motori elettrici, telefoni o calamite), di pulire periodicamente, con gli appositi prodotti, la testina di lettura/registrazione e di farla controllare periodicamente a laboratori attrezzati.
Un ultimo consiglio. Non tenere le cassette o il registratore vicino al televisore: è fonte di intensi campi magnetici che possono modificare le registrazioni contenute sul nastro.

Vettori e Matrici

Fino ad ora abbiamo parlato solo di variabili semplici, cioè di variabili che ad un certo istante dell'esecuzione contenevano un solo valore. Spesse volte, però, risulta essere comodo (od addirittura necessario) poter trattare degli insiemi di dati e variabili legati tra loro da qualche fattore in comune: per esempio i giorni della settimana, i mesi dell'anno o i cognomi dei partecipanti ad una gara di corsa campestre.
La nostra mente è capace di trattare nello stesso momento grandi quantità di informazioni, proprio perché organizza in strutture più ampie tutti i termini associabili tra loro.
Nessuno, per esempio, si ricorda le lettere dell'alfabeto come una serie di simboli senza alcuna relazione: abbiamo piuttosto memorizzato tutte le lettere in un'unica lista cominciante con A. Se per caso ti sorge qualche dubbio, prova ad elencare le lettere dell'alfabeto - partendo dalla Z - altrettanto rapidamente di quanto fai pronunciandole nell'ordine usuale: accantonerai sicuramente ogni perplessità!
Anche i computer sono stati quindi progettati e costruiti per manipolare grandi quantità di variabili semplici, raggruppandole per similitudine in strutture di dati più ampie. Tali strutture prendono il nome di vettori.
Un vettore (o array, o tabella) è perciò una collezione ordinata di variabili - alle quali è possibile riferirsi utilizzando uno stesso nome - ciascuna rappresentante un dato od un valore correlato con gli altri attraverso qualche legame logico o relazionale.
Le singole variabili del vettore vengono chiamate elementi dell'array e sono distinte le une dalle altre mediante il loro numero di posizione all'interno del vettore stesso, che proprio per questa ragione viene anche chiamato indice o subscritto.
Un array può essere di qualsiasi tipo: intero (%), reale, alfanumerico ($); tutti gli elementi appartenenti ad un certo vettore devono però assolutamente essere dello stesso tipo del vettore. Non è quindi consentito mescolare nella stessa tabella dati di tipo differente (e qualsiasi tentativo in proposito porterebbe inevitabilmente alla visualizzazione di un messaggio di errore da parte dell'interprete BASIC).
Il nome di un array può essere un qualunque nome ammesso dal BASIC (segue cioè le stesse regole di definizione delle variabili semplici); è tuttavia necessario che sia sempre seguito da un paio di parentesi. All'interno delle parentesi va infatti messo l'indice corrispondente alla posizione dell'elemento specificato.
Così, un'espressione del tipo

```
PRINT A(17)
```

significa: stampa il diciottesimo elemento appartenente al vettore A.

DIM

Prima che sia possibile utilizzarlo, un array deve però essere "dichiarato": occorre cioè avvisare l'elaboratore di riservare nella memoria un certo numero di cellette contigue (una di seguito all'altra) per potervi memorizzare tutti gli elementi. L'unità centrale non può infatti sapere in anticipo quanto spazio della memoria tu le richiederai. Per creare una tabella è quindi necessario specificare preventivamente il numero di elementi che ne faranno parte. Ciò può essere fatto utilizzando l'istruzione DIM (abbreviazione di DIMensiona): mediante essa si definisce perciò la grandezza dell'array e si riserva nella memoria spazio sufficiente.
Attenzione, però: una volta che il comando DIM sarà stato impartito la dimensione dell'array sarà fissata ed il computer non ti permetterà più di modificarla.

Esempi

```
DIM T(14)
```

crea un array di 15 variabili reali chiamato T. Perché 15 variabili, visto che il numero tra parentesi è 14? Molto semplice: gli elementi vengono numerati partendo dallo zero. Si avranno quindi complessivamente le 15 variabili.

```
T(0), T(1), T(2), T(3), ........, T(12), T(13), T(14)
```

Al momento della DIM ai vari elementi verrà attribuito valore nullo, cioè zero nel caso di array numerici e stringa nulla, per gli array alfanumerici.
Una volta che il vettore sarà stato definito potrai utilizzarne gli elementi come qualsiasi altra variabile.

A(13) = 5.3	Assegna al quattordicesimo elemento della tabella il valore 5.3.
F(19) = F(19) – 3.17	Diminuisce di 3.17 il valore contenuto nella ventesima variabile del vettore F.
C$ (3) = "marzo"	Assegna al quarto elemento dell'array C$ (di tipo stringa) il valore "marzo".

È possibile, anziché utilizzare un numero, specificare l'indice mediante una variabile od un'espressione. In tal caso si dovrà fare particolare attenzione che in nessuna circostanza questa variabile o questa espressione assuma valori non leciti (maggiore della dimensione massima dell'array o minore di zero).

```
LET K = 6
LET V(K) = 73
```

Assegna al settimo elemento del vettore V il valore 73
La possibilità di rappresentare l'indice con variabili consente per esempio di scandire tutto un array, utilizzando un ciclo:

```
 10 DIM V$ (20)
 20 FOR I = 0 TO 20
 30 PRINT "   STRINGA N."; I; : INPUTS$
 40 LET V$ (I) = S$
 50 NEXT I
 60 REM .....
 70 REM .....
100 PRINT "  "
110 FOR I = 0 TO 20
120 PRINT V$ (I)
130 NEXT I
140 REM .....
```

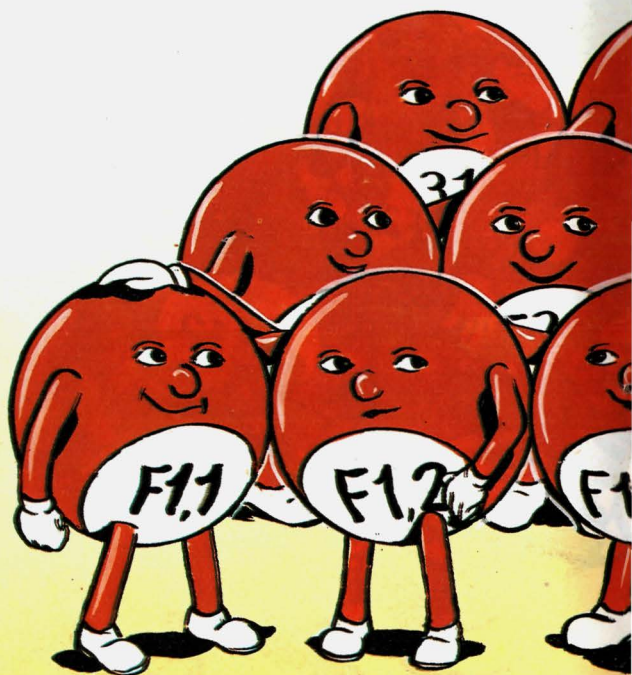
Qualora l'indice sia un numero reale, viene effettuato un troncamento e si considera solo la parte intera:

R (2.8) equivale a R(2)

Le tabelle non sono comunque limitate ad un solo indice; si possono infatti avere anche array a due, tre o più indici. Fisicamente nulla cambia all'interno della memoria: è soltanto (come d'altra parte accadeva prima) una rappresentazione più vicina al modo di ragionare delle persone. Se pensi per esempio alla tavola pitagorica

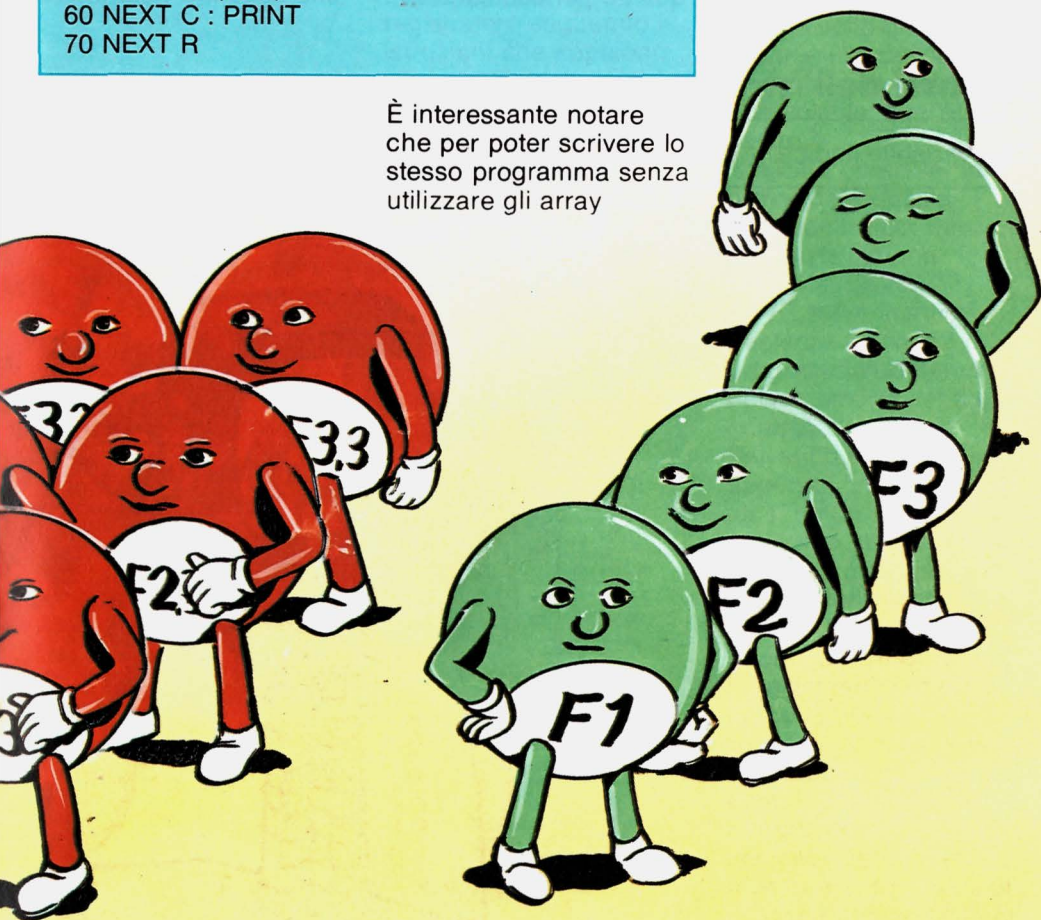
non hai di sicuro nella tua mente una disposizione dei numeri secondo una struttura mono-dimensionale; immagini piuttosto una specie di scacchiera (o, se preferisci, di foglio quadrettato), dove ciascun numero occupa una casella: una rappresentazione, cioè, bi-dimensionale.
Gli array a due indici sono di uso così frequente che posseggono un nome speciale: matrici.
Il BASIC permette di creare matrici mediante il dimensionamento di array a due indici (separati da una virgola): il primo relativo alle righe della matrice ed il secondo alle colonne.
Un'istruzione come

```
DIM Y$ (13, 10)
```

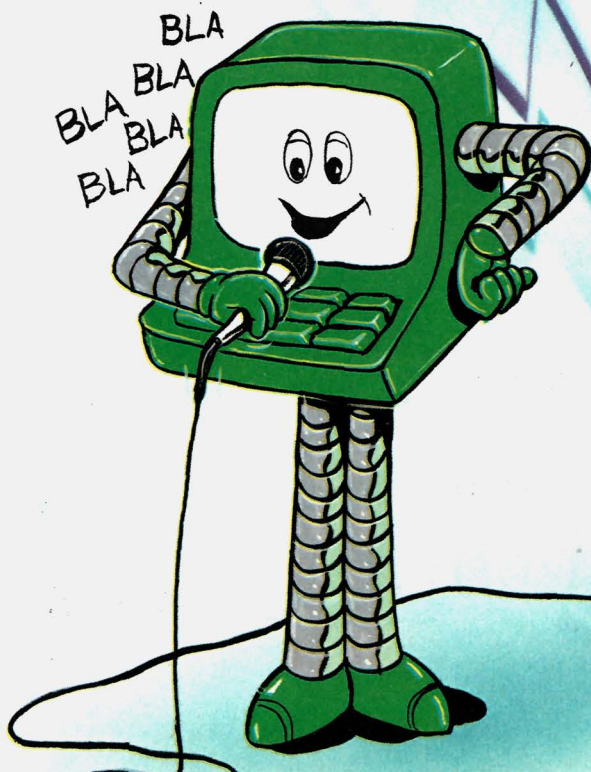
riserva pertanto nella memoria del computer uno spazio sufficiente a contenere le 14 x 11 (= 154) variabili di tipo stringa, che posssiamo immaginare disposte secondo 14 righe e 11 colonne.
Bisogna comunque fare molta attenzione a distinguere righe e colonne: l'elemento Y$ (4,3) occupa infatti una ben differente collocazione nella matrice dell'elemento Y$ (3,4), per quanto sia possibile che in un certo momento gli elementi contengano lo stesso valore.

LINGUAGGIO

Il seguente, breve programma carica nella memoria la tavola di Pitagora:

```
10 DIM P (9, 9) : REM MATRICE DI PITAGORA
20 FOR R = 0 TO 9 : REM R STA PER RIGHE
30 FOR C = 0 TO 9 : REM C STA PER COLONNE
40 LET P (R, C) = (R + 1) * (C + 1) : IF P (R, C)
   < 10 THEN PRINT " ";
50 PRINT P (R, C);
60 NEXT C : PRINT
70 NEXT R
```

È interessante notare che per poter scrivere lo stesso programma senza utilizzare gli array sarebbe stato necessario utilizzare ben 100 variabili semplici, trascinandosi dietro tutti i problemi e gli svantaggi conseguenti.
Si possono dimensionare anche array a tre indici: gli elementi formeranno

allora una struttura tri-dimensionale, che possiamo raffigurare come un parallelepipedo nello spazio.
Dimensioni oltre la terza esulano invece dalla nostra capacità di rappresentazione visiva: proprio per questa ragione il loro uso (per quanto perfettamente consentito in BASIC) è assai raro e sconsigliabile; risulta infatti difficile avere chiaro nella mente questo tipo di modelli.
Nella parte relativa alla programmazione applicheremo ed approfondiremo ulteriormente tutto ciò che abbiamo visto finora.

Sintassi dell'istruzione

DIM variabile (indice1 [,indice2] [,...]) [,variabile (indice1 [,indice2] [,...])]

SAVE

Il tuo VIC 20 riconosce *semplici comandi per* trovare programmi da e su memorie di massa: grazie a questi comandi, *cioè, è possibile* istaurare un colloquio tra computer e periferiche di memoria. (Limitiamo per ora la nostra attenzione al "comune" registratore; ripromettendoci di *riprendere questi* comandi in una fase successiva allorché analizzeremo altre *periferiche di memoria).* Occorre solo premere qualche tasto del registratore seguendo le istruzioni che appaiono via via sullo schermo. Vediamo dunque quali sono queste istruzioni *partendo dalla* fondamentale (per i programmatori, ben inteso, e non per i "caricatori-dipendenti"). Come ben sai la memoria centrale del tuo VIC 20 perde, se priva di alimentazione, ogni contenuto. È necessario, quindi, per non digitare tutte le volte un programma salvarlo su una memoria esterna non volatile, qual'è il nastro.

Il BASIC, come sempre, provvede a questa importantissima funzione con un'istruzione apposita: SAVE.

SAVE serve infatti per trasferire su nastro il *programma contenuto* nella memoria del tuo computer.

Salvando (to save, appunto) un programma gli si da un nome (massimo 16 caratteri) in modo da poter poi riconoscerlo e eseguire successivamente altre operazioni su di lui.

Il trasferimento dei dati da computer a registratore (e viceversa) avviene in modo seriale, cioè un bit alla volta, ricopiando semplicemente i dati in memoria e inviandoli al nastro a una velocità di 300 baud (bit/sec).

Puoi ricavare, perciò, dal tempo di salvataggio (o caricamento come vedremo poi) la lunghezza del programma da salvare (da caricare) senza che tu debba fare nulla, il tuo computer, o meglio il suo sistema operativo, provvede a porre all'inizio della registrazione l'header, cioè i dati necessari per il riconoscimento del programma da parte della CPU: il nome del programma, la lunghezza in byte e il suo collocamento di partenza nella memoria. Vediamo cosa devi fare operativamente per salvare un programma. Una volta dato SAVE il computer risponderà inviandoti il messaggio

PRESS RECORD AND PLAY ON TAPE

Attento a premere prima RECORD e poi PLAY. Se premi, infatti, anche solo leggermente un tasto del registratore, inizia l'invio dei dati senza che questi siano registrati, perdendoli. Niente paura: in tal caso premi il tasto RUN/STOP e riprendi da capo la sequenza.

Alla fine del salvataggio ti apparirà

READY

E ora un po' di consigli dettati dall'esperienza:

- non pensare di abusare mai abbastanza di SAVE. Man mano che procedi nella stesura di un programma, salvalo! Un contatto elettrico difettoso, il solito amico ..., la mancanza di elettricità possono compromettere ore e ore di lavoro;
- prima di registrare accertati che la parte di nastro su cui ti accingi a salvare un programma non ne contenga un altro (o parte di programma) che ti serve: lo perderesti per sempre;
- usa più cassette (preferibilmente corte): alcune di lavoro, altre per i programmi che vuoi assolutamente conservare. Se non vuoi correre rischi di sovraincisione togli le linguette poste sul retro della cassetta.

Se seguirai questi consigli, ti eviterai tutti

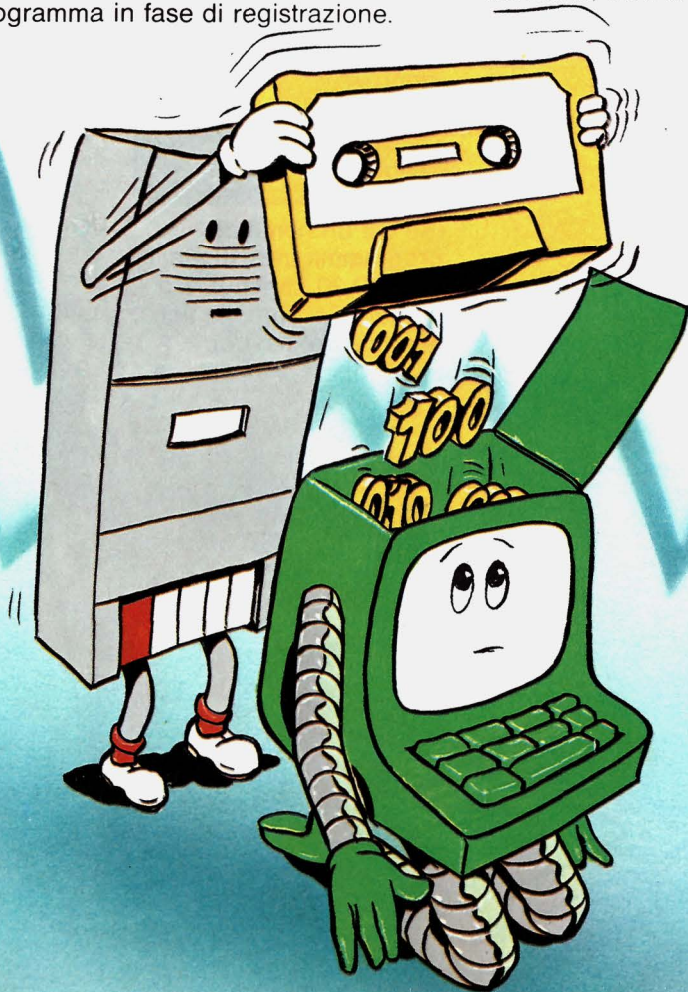
gli inconvenienti che ogni programmatore agli inizi (compresi noi) ha incontrato.

Sintassi dell'istruzione/comando

SAVE "nome del programma"

il nome del programma è il nome che è stato assegnato al programma in fase di registrazione.

LOAD

Una volta salvato un programma, o acquistata una cassetta. commerciale, si pone il problema (si fa per dire visto che stai seguendo questo corso) di caricarla, cioè di

eseguire l'operazione inversa di SAVE: trasferire il programma dalla memoria di massa alla memoria centrale. L'istruzione, o comando, usata per questo trasferimento è LOAD; il reciproco di SAVE.

LOAD può essere seguito dal nome del programma oppure da ... nulla.
Vediamo ora cosa operativamente succede. Quando dai il comando LOAD, inizia un colloquio con il computer. Esso ti risponde via via:

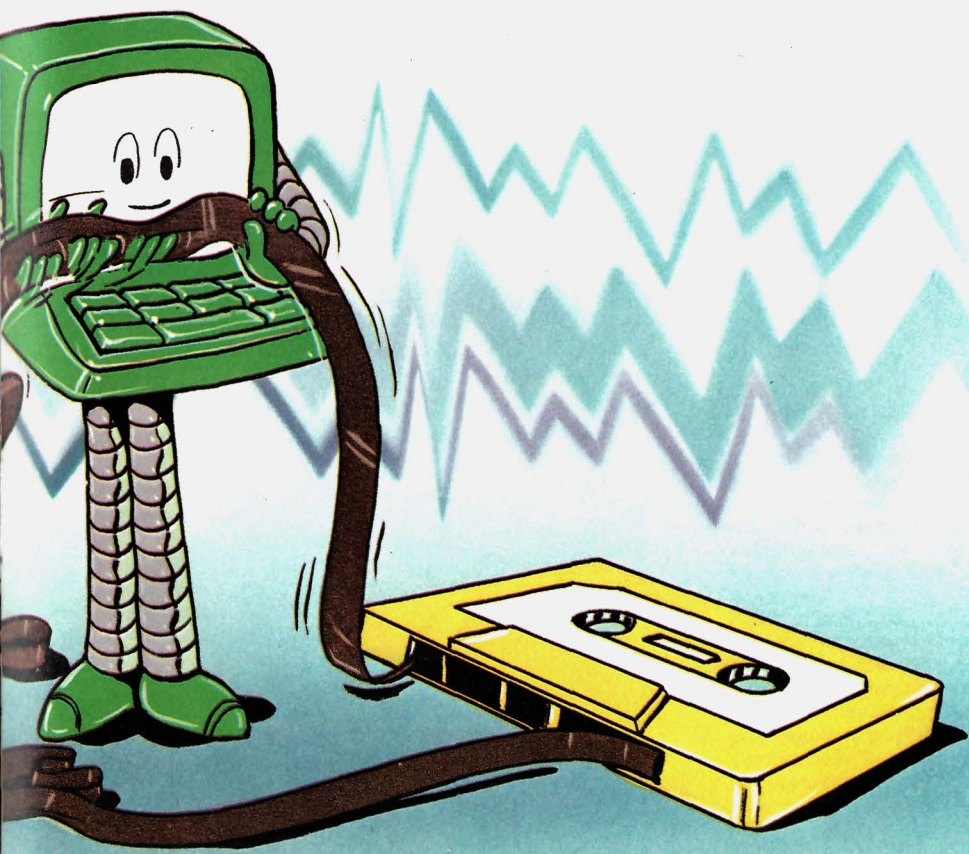
```
PRESS PLAY ON TAPE
SEARCHING
FOUND
nome del programma
LOADING
READY
```

(Se usi un nome di programma inesistente, il tuo VIC 20 continuerà a cercarlo inutilmente fino alla fine del nastro).

A questo punto dai RUN. Il tuo programma girerà (detto in termini tecnici). Puoi anche caricare automaticamente il programma premendo al posto di LOAD, SHIFT-RUN/STOP: il primo programma che il tuo VIC 20 incontra viene "lanciato" automaticamente. (Altro termine gergale per indicare che è fatto partire dopo essere stato caricato).
Seguendo i consigli

forniti con SAVE sarete sicuri di trovare subito e di caricare con tutta tranquillità il programma voluto.
Un'ultima avvertenza. Come nel salvataggio anche nel caricamento occorre attenzione: LOAD infatti cancella nella memoria centrale il vecchio programma e le sue variabili. Come sempre prima di procedere una rapida riflessione su quello che si sta per fare non guasta.

Sintassi dell'istruzione/comando

LOAD "nome programma"

VERIFY

Per controllare che dopo un comando SAVE tutto sia andato per il verso giusto (cioè che non vi siano state anomalie di registrazione) è possibile confrontare il programma nella memoria con il corrispondente programma memorizzato su nastro. Se il confronto rileva delle differenze, si ha l'insorgere del messaggio di errore

LOAD ERROR

Per effettuare un confronto, però, occorre indicare al tuo VIC 20 cosa confrontare il programma in memoria centrale.
Per questo VERIFY viene seguita dal nome del programma di cui verificare il salvataggio. Senza alcun nome provocherà invece il confronto tra il programma in memoria ed il primo programma incontrato sulla cassetta.

Sintassi dell'istruzione

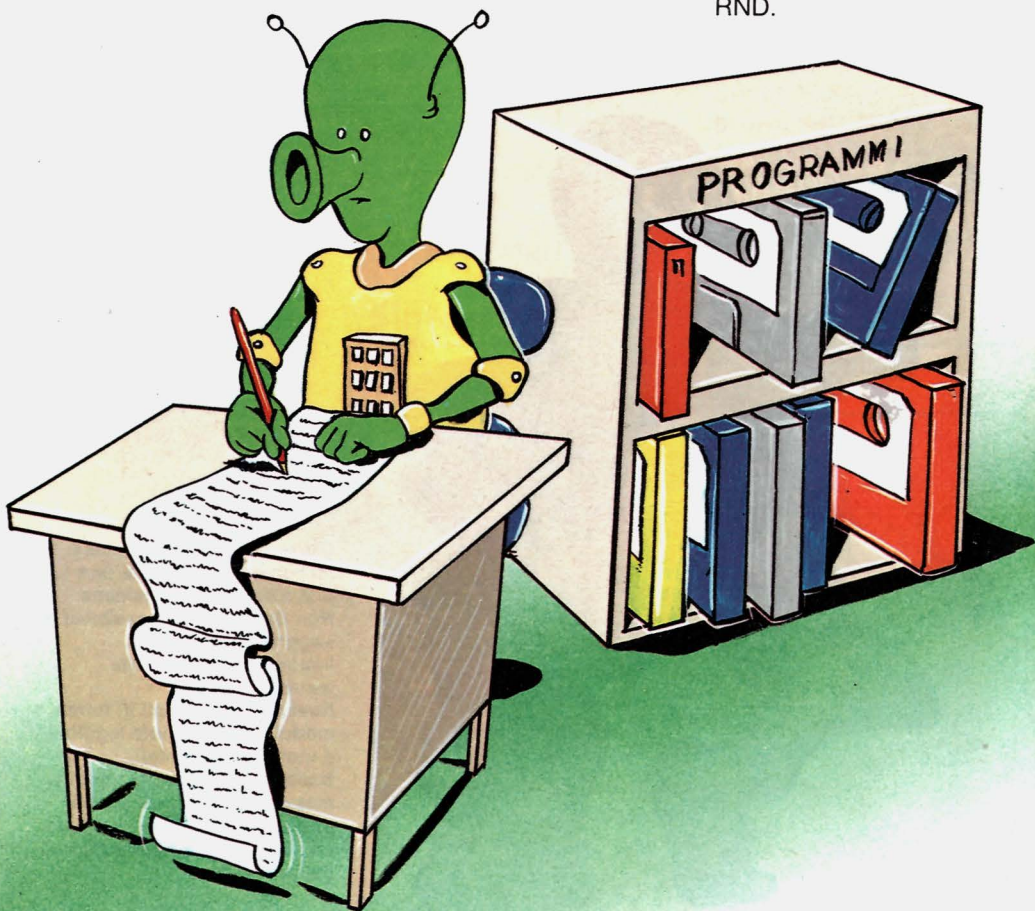
VERIFY "nome del programma"

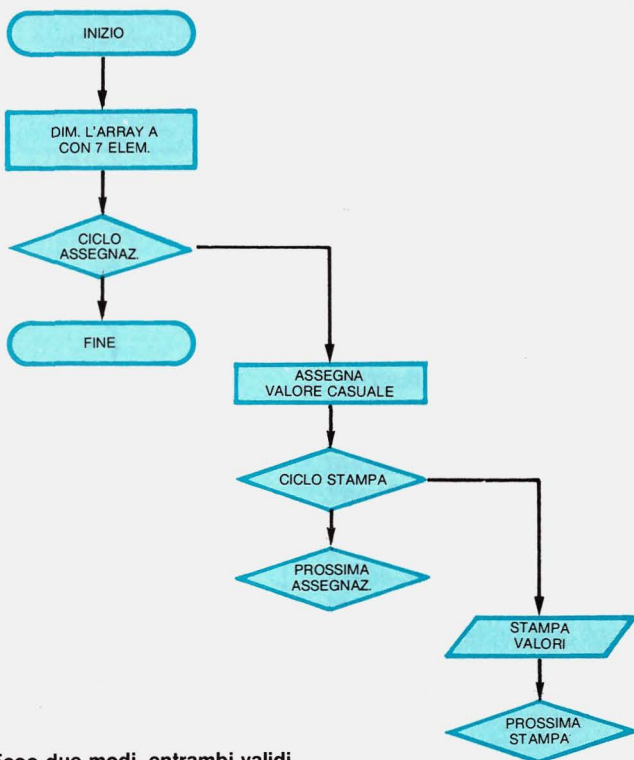
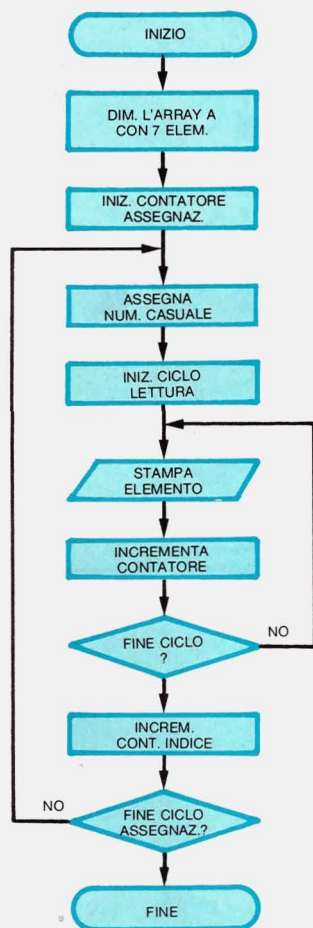
LINGUAGGIO

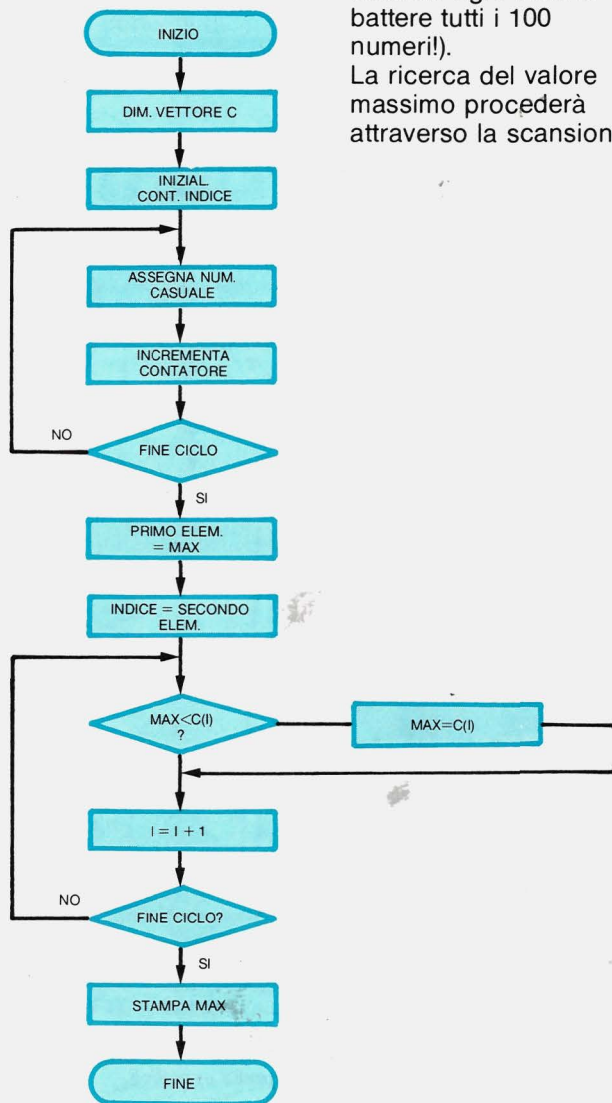
L'errore è sempre in attesa: malignamente aspetta una tua distrazione, anche minima.
Non dargli mai la possibilità di sogghignare.
Impara la sintassi delle istruzioni.
Realizza programmi in forma modulare, facilmente leggibili, e molto documentati.
Nella digitazione, poi, metti la massima attenzione: anche una virgola al posto sbagliato, compromette la buona riuscita del programma.

PROGRAMMAZIONE

Uso di vettori e matrici

Il primo programma che oggi esamineremo è un esempio introduttivo all'uso degli array. Adoperando i vettori è infatti indispensabile avere ben chiari nella mente i concetti di elemento, valore ed indice: questo programma dovrebbe perciò aiutarti a risolvere eventuali dubbi ed incertezze.
Ciò che ci proponiamo di fare è molto semplice: definire un array di 7 elementi e memorizzare in ciascuno di essi un valore a caso - compreso tra 0 e 100 - generato dalla funzione RND.

```
10 DIM A (6)
20 FOR I = 0 TO 6
30 LET A (I) = INT (RND (0) * 100)
40 REM VISUALIZZA I VALORI ASSEGNATI
50 FOR J = 0 TO 6 : IF A (J) < 10 THEN PRINT " ";
60 PRINT A (J); "–";
70 NEXT J
80 PRINT
90 NEXT I
```

L'esecuzione di questo programma è resa interessante dalla ripetuta stampa sullo schermo dei valori contenuti nel vettore. All'interno del ciclo FOR principale (quello, per intenderci, che inizia alla linea 20) è stato infatti

Ecco due modi, entrambi validi e corretti, per rappresentare il medesimo problema. Il primo più sequenziale il secondo più strutturato. Esaminali entrambi e giudica tu quale, a tuo avviso, sia il più consono alla esigenza specifica.

inserito un secondo ciclo (linee 50-70) che visualizza sullo schermo i valori in quel momento presenti nell'array.
Man mano che la variabile indice verrà incrementata vedrai come gli zeri - contenuti inizialmente nel vettore in seguito alla DIM - saranno sostituiti dai numeri casuali. Sul tuo video avrai quindi una copia fedele di quanto sarà successo, durante il programma, all'interno della memoria del tuo computer.
Come puoi notare, è perfettamente lecito utilizzare due indici differenti (nel nostro caso I e J) per riferirsi ad uno stesso elemento dell'array: l'importante è solo che rispettino i limiti di definizione del vettore.
Come secondo esempio vediamo un programma che ricerca il massimo tra un certo gruppo di numeri (per esempio 100).
Questi numeri devono venir memorizzati all'interno di un array; per nostra comodità automatizzeremo l'inserimento dei valori con un ciclo FOR e la funzione RND (sarebbe infatti troppo noioso e frustrante essere costretti ogni volta a battere tutti i 100 numeri!).
La ricerca del valore massimo procederà attraverso la scansione

di ogni singolo elemento del vettore (assegnando alla variabile MAX il valore massimo incontrato sino a quel momento) fintanto che l'array non sarà stato ispezionato per intero.

```
 10 LET J = 99
 20 DIM C (J)
 30 REM CARICAMENTO DELL'ARRAY
 40 FOR I = 0 TO J
 50 LET C (I) = INT (RND (0) * 100)
 60 NEXT I
 70 REM RICERCA DEL NUMERO MASSIMO
 80 LET MAX = C (0) : REM PRIMO ELEMENTO = MAX
 90 FOR I = I TO J
100 IF MAX < C (I) THEN MAX = C (I)
110 NEXT I
120 PRINT "♥Q IL VALORE MASSIMO È"; MAX; "Q"
130 PRINT "ECCO I NUMERI CONTENUTI NELL'ARRAY : Q"
140 FOR I = 0 TO J : IF C (I) < 10 THEN PRINT " ";
150 PRINT C (I); : NEXT I
160 END
```

Prova a modificare il programma, cercando di trovare - oltre al valore massimo - anche il valore minimo. Può anche essere interessante rilevare il drastico aumento del tempo di esecuzione conseguente ad un incremento (per esempio da 100 a 500) della dimensione dell'array. Non hai che da sbizzarrirti!
Il terzo ed ultimo esempio della nostra lezione è la soluzione del seguente problema: data una lista di numeri

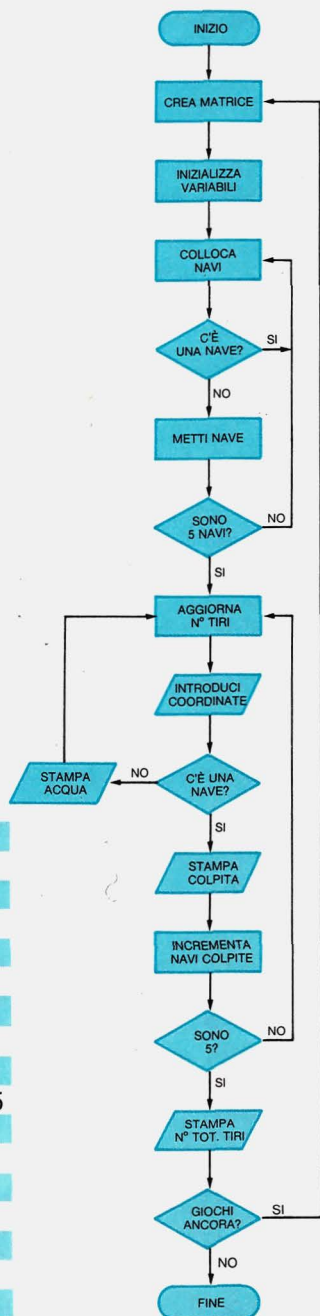
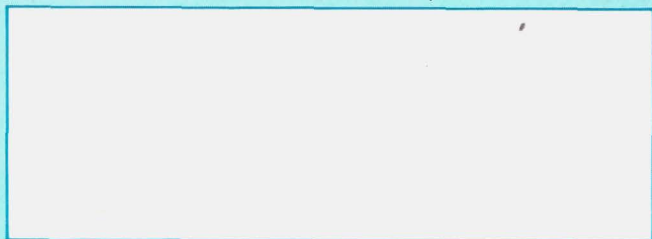
interi trovare quanti numeri dispari essa contiene e determinarne le posizioni occupate. Come prima, l'inserimento dei numeri è affidato a RND. Cerca nuovamente di apportare modifiche e miglioramenti; potresti per esempio calcolare la somma degli elementi nei due array, oppure cercare in quali posizioni si trovano numeri compresi in un certo intervallo, od ancora (anche se questo lo vedremo insieme in una delle prossime lezioni) ordinare in ordine crescente o decrescente i vari

```
10 REM IL VETTORE A CONTIENE I NUMERI, P LE POSIZIONI DEI
   NUMERI DISPARI
20 INPUT "QUANTI SONO I NUMERI?"; NUM: LET NUM = NUM - 1
30 DIM A (NUM), P (NUM)
40 FOR I = 0 TO NUM
50 LET A (I) = INT (RND (0) * 1000) : PRINT I, A (I): REM
   I NUMERI SARANNO COMPRESI TRA 0 E 999
60 NEXT I
70 REM INIZIALIZZA IL CONTATORE DEI NUMERI DISPARI
80 LET K = 0
90 REM CERCA I NUMERI DISPARI
100 FOR I = 0 TO NUM
110 REM DIVIDI PER 2 IL NUMERO E SE C'È RESTO, VUOL DIRE CHE
    È DISPARI
120 LET B = A (I)/2
130 REM PRENDE LA PARTE INTERA DELLA DIVISIONE
140 LET C = INT (B)
150 REM SE LA PARTE INTERA DEL RISULTATO DELLA DIVISIONE
    È UGUALE AL RISULTATO STESSO, VUOL DIRE CHE NON
    ESISTE PARTE DECIMALE E QUINDI IL NUMERO È PARI
180 IF B = C THEN GOTO 240
190 REM IL NUMERO È DISPARI, QUINDI METTI LA POSIZIONE
    DEL NUMERO NEL VETTORE POSIZIONI
210 LET P (K) = I
220 REM INCREMENTA IL CONTATORE DEI NUMERI DISPARI
230 LET K = K + 1
240 NEXT I
250 REM STAMPA LA QUANTITA' DEI NUMERI DISPARI E LE LORO
    POSIZIONI
270 PRINT "I NUMERI DISPARI SONO"; K
280 PRINT "I NUMERI DISPARI OCCUPANO LE POSIZIONI:";
290 FOR I = 0 TO K - 1
300 PRINT P (I); "-";
310 NEXT I
320 END
```

elementi, inizialmente disposti a caso. Qualsiasi tentativo non potrà che incrementare la tua confidenza con gli array o vettori e le matrici.

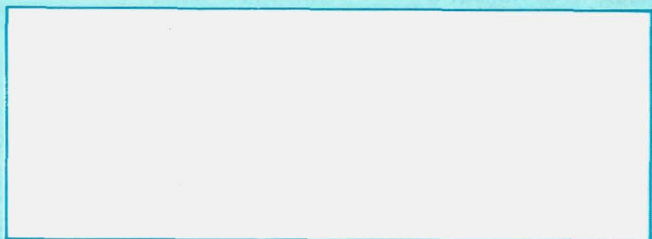
Battaglia navale

Il gioco della battaglia navale è talmente conosciuto da non aver bisogno di nessuna spiegazione. Interessante è invece istruire il computer a gestirlo: si tratta cioè di affidare al tuo VIC 20 il compito di predisporre il campo di battaglia, il collocamento casuale delle navi, il controllo delle cannonate.
DIM avrà la funzione di costruire la tabella delle caselle che costituiscono la zona di mare interessata al combattimento mentre RND provvederà a disporre le navi nemiche in posizioni sconosciute e imprevedibili.
Provando a giocare ti renderai conto di quanto la difficoltà dipenda dalla fortuna e dalle dimensioni della matrice. Prova a ridurla. Inventa anche un modo di farti dire dal computer dove si trovano le navi che non riesci a colpire.

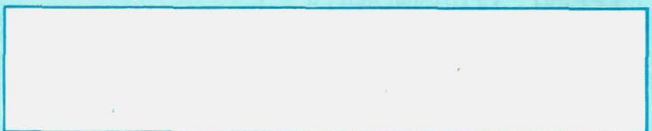
```
10 DIM A (5,5) : LET T=0 : LET B=0
15 FOR N = 1 TO 5
20 LET C = INT (RND (1) * 6)
25 LET R = INT (RND (1) * 6)
30 IF A (C,R) = 1THEN GOTO 20
35 LET A (C,R) = 1 : NEXT
40 LET T = T + 1
45 PRINT " ◘ CANNONATA" ; T
50 INPUT "COORDINATE" ; C,R
55 IFA (C,R) = 0 THEN PRINT" ◘ ACQUA": GOTO 65
60 PRINT "◘ COLPITA!" : LET
     B = B + 1 : LET A (C,R) = 0
65 IF B < 5 THEN GOTO 40
70 PRINT " ◘ CANNONATE" ; T
75 INPUT "RIPROVI" ; R$
80 IF R$ = "S" THEN RUN
85 END.
```

```
10 PRINT " ♥ "
20 DIM A (39)
30 FOR I = 0 TO 39
40 LET A (I) = INT (...
50 NEXT I
60 FOR I = 0 TO 39
70 PRINT A (I)
80 NEXT I
```

— Completa la linea 40 in modo da assegnare ad ogni elemento del vettore un numero intero casuale compreso tra 0 e 499.

— Modifica il Loop di stampa alla linea 60 per stampare il vettore al contrario.

Caccia all'errore

```
10 PRINT " ♥ "
20 DIM D (11,9)
30 FOR I = 0 TO 11
40 INPUT D (I)
50 FOR J = 0 TO 9
60 INPUT D (J)
70 NEXT I
80 NEXT J
90 REM ORRORE!
```

Concentrati sul loop di INPUT e pensa alla nidificazione dei cicli FOR. Non è possibile inoltre assegnare una variabile a due dimensione, in due tempi!

Matrice ad elementi sconosciuti

```
10 INPUT "PRIMA DIMENSIONE ="; P
20 INPUT "SECONDA DIMENSIONE ="; S
30 P = P — 1 : S = S — 1 : DIM (P,S)
40 FOR I = 0 TO P
50 FOR J = 0 TO S
60 INPUT C (P,S)
70 NEXT J
80 NEXT I
90 PRINT "MATRICE COMPLETA"
```

Attenzione a non esagerare con le dimensioni perchè il numero degli elementi da introdurre è dato dal prodotto degli indici. Aggiungi delle linee in grado di ristampare i valori introdotti.